

L Number	Hits	Search Text	DB	Time stamp
1	73	(multiprocessor or (multiple adj processor)) and parallel\$4 and heaps!	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/18 16:03
2	24	((multiprocessor or (multiple adj processor)) and parallel\$4 and heaps!) and (collect\$4 or gather\$4) near garbage	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/18 16:06
3	0	(((multiprocessor or (multiple adj processor)) and parallel\$4 and heaps!) and (collect\$4 or gather\$4) near garbage) and (multiprocessor or (multiple adj processor) or processors!) with parallel\$4 with heap	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/18 16:04
4	6	(((multiprocessor or (multiple adj processor)) and parallel\$4 and heaps!) and (collect\$4 or gather\$4) near garbage) and (multiprocessor or (multiple adj processor) or processor) with heap	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/18 16:04
5	0	(((multiprocessor or (multiple adj processor)) and parallel\$4 and heaps!) and (collect\$4 or gather\$4) near garbage) and (multiprocessor or (multiple adj processor) or processor) with parallel\$4 with heap	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/18 16:06
6	3838	(collect\$4 or gather\$4) near garbage	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/18 16:06
7	0	((collect\$4 or gather\$4) near garbage) and (multiprocessor or (multiple adj processor) or processor) with parallel\$4 with heap	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/18 16:07
8	29	((collect\$4 or gather\$4) near garbage) and (multiprocessor or (multiple adj processor) or processor) near6 heap	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/18 16:09
9	9	((collect\$4 or gather\$4) near garbage) and (multiprocessor or (multiple adj processor) or processor) near6 heap) and (multiprocessor or (multiple adj processor) or processor) near6 parallel\$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/18 16:07
10	424	((collect\$4 or gather\$4) near garbage) and (multiprocessor or (multiple adj processor))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/18 16:09
11	13	((collect\$4 or gather\$4) near garbage) and (multiprocessor or (multiple adj processor))) and processor near4 heap	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/18 16:10
-	79473	707/\$.ccls. or 345/\$.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/12 14:16
-	21483	(707/\$.ccls. or 345/\$.ccls.) and (database or document or (data stream)) same (analy\$6 or filter\$4 or classif\$6 or find\$4)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/12 14:15

-	1607	345/764,853,854,855.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/12 14:15
-	706	345/764,853,854,855.ccls. and (database or document or (data stream)) same (analy\$6 or filter\$4 or classif\$6 or find\$4)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/12 14:15
-	365	345/764,853,854,855.ccls. and (database or document or (data near stream)) same (analy\$6 or filter\$4 or classif\$6 or find\$4)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/12 14:16
-	126	((345/764,853,854,855.ccls. and (database or document or (data near stream)) same (analy\$6 or filter\$4 or classif\$6 or find\$4)) and (707/\$.ccls. or 715/\$.ccls.))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/12 14:16
-	126	((345/764,853,854,855.ccls. and (database or document or (data near stream)) same (analy\$6 or filter\$4 or classif\$6 or find\$4)) and (707/\$.ccls. or 715/\$.ccls.)) and (display\$4 or visual\$6 or view\$4 or present\$5)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/12/12 14:17



[> home](#) [> about](#) [> feedback](#) [> login](#)

US Patent & Trademark Office



Try the *new* Portal design

Give us your opinion after using it.

Search Results

Search Results for: **[heap <near/6> processor<AND>((heaps<AND>(("garbage collection" <paragraph> (multi-processor or multiprocessor or "multiple processors"))))]**

Found **16** of **124,998** searched.

Search within Results



[> Advanced Search](#)

[> Search Help/Tips](#)

Sort by: **Title** **Publication** **Publication Date** **Score** Binder

Results 1 - 16 of 16 **short listing**

- 1** **Hoard: a scalable memory allocator for multithreaded applications** 100%

Emery D. Berger , Kathryn S. McKinley , Robert D. Blumofe , Paul R. Wilson

Proceedings of the ninth international conference on Architectural support for programming languages and operating systems November 2000

Volume 28 , 34 Issue 5 , 5

Parallel, multithreaded C and C++ programs such as web servers, database managers, news servers, and scientific applications are becoming increasingly prevalent. For these applications, the memory allocator is often a bottleneck that severely limits program performance and scalability on multiprocessor systems. Previous allocators suffer from problems that include poor performance and scalability, and heap organizations that introduce false sharing. Worse, many allocators exhibit a dramatic incr ...
- 2** **Hoard: a scalable memory allocator for multithreaded applications** 100%

Emery D. Berger , Kathryn S. McKinley , Robert D. Blumofe , Paul R. Wilson

ACM SIGPLAN Notices November 2000

Volume 35 Issue 11

Parallel, multithreaded C and C++ programs such as web servers, database managers, news servers, and scientific applications are becoming increasingly prevalent. For these applications, the memory allocator is often a bottleneck that severely limits program performance and scalability on multiprocessor systems. Previous allocators suffer from problems that include poor performance and scalability, and heap organizations that introduce false sharing. Worse, many allocators exhibit a dramatic incr ...
- 3** **A space-efficient parallel garbage compaction algorithm** 100%

Wolfgang Küchlin

Proceedings of the 5th international conference on Supercomputing June 1991

4 Creating and preserving locality of java applications at allocation and garbage collection times 99%



Yefim Shuf , Manish Gupta , Hubertus Franke , Andrew Appel , Jaswinder Pal Singh
ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications November 2002

Volume 37 Issue 11

The growing gap between processor and memory speeds is motivating the need for optimization strategies that improve data locality. A major challenge is to devise techniques suitable for pointer-intensive applications. This paper presents two techniques aimed at improving the memory behavior of pointer-intensive applications with dynamic memory allocation, such as those written in Java. First, we present an allocation time object placement technique based on the recently introduced notion of p ...

5 Mostly lock-free malloc 99%



Dave Dice , Alex Garthwaite
ACM SIGPLAN Notices , Proceedings of the third international symposium on Memory management June 2002

Volume 38 Issue 2 supplement

Modern multithreaded applications, such as application servers and database engines, can severely stress the performance of user-level memory allocators like the ubiquitous malloc subsystem. Such allocators can prove to be a major scalability impediment for the applications that use them, particularly for applications with large numbers of threads running on high-order multiprocessor systems. This paper introduces Multi-Processor Restartable Critical Sections, or MP-RCS. MP-RCS permits user-level ...

6 A scalable mark-sweep garbage collector on large-scale shared-memory machines 99%



Toshio Endo , Kenjiro Taura , Akinori Yonezawa
Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM) November 1997

This work describes implementation of a mark-sweep garbage collector (GC) for shared-memory machines and reports its performance. It is a simple "parallel" collector in which all processors cooperatively traverse objects in the global shared heap. The collector stops the application program during a collection and assumes a uniform access cost to all locations in the shared heap. Implementation is based on the Boehm-Demers-Weiser conservative GC (Boehm GC). Experiments have been done on Ultra ...

7 MULTILISP: a language for concurrent symbolic computation 99%



Robert H. Halstead
ACM Transactions on Programming Languages and Systems (TOPLAS) October 1985

Volume 7 Issue 4

Multilisp is a version of the Lisp dialect Scheme extended with constructs for parallel execution. Like Scheme, Multilisp is oriented toward symbolic computation. Unlike some parallel programming languages, Multilisp incorporates constructs for causing side effects and for explicitly introducing parallelism. The potential complexity of dealing with side effects in a parallel context is mitigated by the nature of the parallelism constructs and by support for abstract data types: a recommende ...

8 Concurrent compacting garbage collection of a persistent heap 99%



James O'Toole , Scott Nettles , David Gifford

ACM SIGOPS Operating Systems Review , Proceedings of the fourteenth ACM symposium on Operating systems principles December 1993

Volume 27 Issue 5

We describe a replicating garbage collector for a persistent heap. The garbage collector cooperates with a transaction manager to provide safe and efficient transactional storage management. Clients read and write the heap in primary memory and can commit or abort their write operations. When write operations are committed they are preserved in stable storage and survive system failures. Clients can freely access the heap during garbage collection because the collector concurrently builds a comp ...

9 Thread-specific heaps for multi-threaded programs 98%



Bjarne Steensgaard

ACM SIGPLAN Notices , Proceedings of the second international symposium on Memory management October 2000

Volume 36 Issue 1

Garbage collection for a multi-threaded program typically involves either stopping all threads while doing the collection or involves copious amounts of synchronization between threads. However, a lot of data is only ever visible to a single thread, and such data should ideally be collected without involving other threads.

Given an escape analysis, a memory management system may allocate thread-specific data in thread-specific heaps and allocate shared data in a shared heap. Garbage c ...

10 Heap architectures for concurrent languages using message passing 97%



Erik Johansson , Konstantinos Sagonas , Jesper Wilhelmsson

ACM SIGPLAN Notices , Proceedings of the third international symposium on Memory management June 2002

Volume 38 Issue 2 supplement

We discuss alternative heap architectures for languages that rely on automatic memory management and implement concurrency through asynchronous message passing. We describe how interprocess communication and garbage collection happens in each architecture, and extensively discuss the tradeoffs that are involved. In an implementation setting (the Erlang/OTP system) where the rest of the runtime system is unchanged, we present a detailed experimental comparison between these architectures using bo ...

11 Portable, unobtrusive garbage collection for multiprocessor systems 97%



Damien Doligez , Georges Gonthier

Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages February 1994

We describe and prove the correctness of a new concurrent mark-and-sweep garbage collection algorithm. This algorithm derives from the classical on-the-fly algorithm from Dijkstra et al. [9]. A distinguishing feature of our algorithm is that it supports multiprocessor environments where the registers of running processes are not readily accessible, without imposing any overhead on the elementary operations of loading a register or reading or initializing a field. Furthermor ...

12 An abstract machine for parallel graph reduction

97%



Lal George

Proceedings of the fourth international conference on Functional programming languages and computer architecture November 1990**13** Design and performance of a coherent cache for parallel logic

97%



programming architectures

A. Goto , A. Matsumoto , E. Tick

ACM SIGARCH Computer Architecture News , Proceedings of the 16th annual international symposium on Computer architecture April 1989

Volume 17 Issue 3

This paper describes the design and performance of a tightly-coupled shared-memory coherent cache optimized for the execution of parallel logic programming architectures. The cache utilizes a copy-back write-allocation protocol having five states and a hardware lock mechanism. Optimizations for logic programming are introduced in four software-controlled memory access commands: direct-write, exclusive-read, read-purge, and read-invalidate. In this paper we describe these operations and pres ...

14 A parallel, real-time garbage collector

95%



Perry Cheng , Guy E. Bluelloch

ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation May 2001

Volume 36 Issue 5

We describe a parallel, real-time garbage collector and present experimental results that demonstrate good scalability and good real-time bounds. The collector is designed for shared-memory multiprocessors and is based on an earlier collector algorithm [2], which provided fixed bounds on the time any thread must pause for collection. However, since our earlier algorithm was designed for simple analysis, it had some impractical features. This paper presents the extensions necessary for a pract ...

15 An architecture for efficient Lisp list access

88%



A. R. Pleszkun , M. J. Thazhuthaveetil

ACM SIGARCH Computer Architecture News , Proceedings of the 13th annual international symposium on Computer architecture June 1986

Volume 14 Issue 2

In this paper, we present a Lisp machine architecture that supports efficient list manipulation. This Lisp architecture is organized as two processing units: a List Processor (LP), that performs all list related operations and manages the list memory, and an Evaluation Processor (EP), that maintains the addressing and control environment. The LP contains a translation table (LPT) that maps a small set of list identifiers into the physical memory addresses of objects. Essentially, the LP and ...

16 Implementation of multilisp: Lisp on a multiprocessor

87%



Robert H. Halstead

Proceedings of the 1984 ACM Symposium on LISP and functional programming August 1984

Multilisp is an extension of Lisp (more specifically, of the Lisp dialect Scheme [15]) with additional operators and additional semantics to deal with parallel execution. It is being implemented on the 32-processor Concert multiprocessor. The current

implementation is complete enough to run the Multilisp compiler itself, and has been run on Concert prototypes including up to four processors. Novel techniques are used for task scheduling and garbage collection. The task sche ...

Results 1 - 16 of 16 short listing

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2003 ACM, Inc.

IEEE HOME | SEARCH IEEE | SHOP | WEB ACCOUNT | CONTACT IEEE



Membership Publications/Services Standards Conferences Careers/Jobs

IEEE Xplore®
RELEASE 1.5Welcome
United States Patent and Trademark Office[Help](#) [FAQ](#) [Terms](#) [IEEE Peer Review](#)[Quick Links](#)

Welcome to IEEE Xplore®

[SEARCH RESULTS](#)[\[PDF Full-Text \(340 KB\)\]](#)[PREVIOUS](#)[NEXT](#)[DOWNLOAD CITATION](#)

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account
- ☐ Access the IEEE Member Digital Library

[Print Format](#)

On multi-threaded list-processing and garbage collection

Kuechlin, W.W. Nevin, N.J.

Dept. of Comput. & Inf. Sci., Ohio State Univ., Columbus, OH;

*This paper appears in: **Parallel and Distributed Processing, 1991.*****Proceedings of the Third IEEE Symposium on**

Meeting Date: 12/02/1991 -12/05/1991

Publication Date: 2-5 Dec 1991

Location: Dallas, TX, USA

On page(s): 894-897

References Cited: 17

IEEE Catalog Number: 91TH0396-2

INSPEC Accession Number: 4368138

Abstract:

The authors discuss the problem of parallel list-processing and garbage collection in an environment based on lightweight processes (threads). Their main insight is that the threads paradigm suggests a heap memory layout and garbage collection technique which is quite different from existing Lisp and Prolog systems. They introduce a hierarchy of fork constructs and a memory structure which supports garbage collection schemes which are local to threads. For example, the new technique of preventive garbage collection can recover all intermediate list memory used by a function at the small expense of copying its output parameters.

Index Terms:

[list processing](#) [storage management](#) [fork constructs](#) [garbage collection](#) [heap memory layout](#) [intermediate list memory](#) [memory structure](#) [multithreaded list processing](#) [parallel list-processing](#)

Documents that cite this document

Select link to view other documents in the database that cite this one.

[SEARCH RESULTS](#)[\[PDF Full-Text \(340 KB\)\]](#)[PREVIOUS](#)[NEXT](#)[DOWNLOAD CITATION](#)

[Home](#) | [Log-out](#) | [Journals](#) | [Conference Proceedings](#) | [Standards](#) | [Search by Author](#) | [Basic Search](#) | [Advanced Search](#) | [Join IEEE](#) | [Web Account](#) | [New this week](#) | [OPAC Linking Information](#) | [Your Feedback](#) | [Technical Support](#) | [Email](#) | [No Robots Please](#) | [Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)

[IEEE HOME](#) | [SEARCH IEEE](#) | [SHOP](#) | [WEB ACCOUNT](#) | [CONTACT IEEE](#)

[Membership](#) | [Publications/Services](#) | [Standards](#) | [Conferences](#) | [Careers/Jobs](#)
IEEE Xplore®
RELEASE 1.5

 Welcome
United States Patent and Trademark Office

[Help](#) | [FAQ](#) | [Terms](#) | [IEEE Peer Review](#)
[Quick Links](#)

Welcome to IEEE Xplore®

[SEARCH RESULTS](#)
[\[PDF Full-Text \(960 KB\)\]](#)
[PREVIOUS](#)
[NEXT](#)
[DOWNLOAD CITATION](#)

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account
- ☐ Access the IEEE Member Digital Library

Print Format

Evaluation of parallel copying garbage collection on shared-memory multiprocessor

Imai, A. Tick, E.

Inst. for New Generation Comput. Technol., Tokyo ;

*This paper appears in: **Parallel and Distributed Systems, IEEE Transactions on***

Publication Date: Sep 1993

On page(s): 1030-1040

Volume: 4, Issue: 9

ISSN: 1045-9219

References Cited: 22

CODEN: ITDSEO

INSPEC Accession Number: 4582750

Abstract:

A parallel copying garbage collection algorithm for symbolic languages executed on shared-memory multiprocessors is proposed. The algorithm is an extension of Baker's sequential algorithm with a novel method of heap allocation to prevent fragmentation and facilitate load distribution during garbage collection. An implementation of the algorithm within a concurrent logic programming system (VLIM), has been evaluated and the results, for a wide selection of benchmarks analyzed here. The authors show 1) how much the algorithm reduces the contention for critical sections during garbage collection, 2) how well the load balancing strategy works and its expected overheads, and 3) the expected speed achieved by the algorithm.

Index Terms:

logic programming parallel algorithms resource allocation shared memory systems storage management VLIM concurrent logic programming system contention fragmentation garbage collection heap allocation load distribution load-balancing parallel copying shared-memory multiprocessor symbolic languages

Documents that cite this document

Select link to view other documents in the database that cite this one.

[SEARCH RESULTS](#)
[\[PDF Full-Text \(960 KB\)\]](#)
[PREVIOUS](#)
[NEXT](#)
[DOWNLOAD CITATION](#)

[No Robots Please](#) | [Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)

Copyright © 2003 IEEE — All rights reserved

IEEE HOME | SEARCH IEEE | SHOP | WEB ACCOUNT | CONTACT IEEE



Membership Publications/Services Standards Conferences Careers/Jobs

IEEE Xplore®
RELEASE 1.5Welcome
United States Patent and Trademark Office[Help](#) [FAQ](#) [Terms](#) [IEEE Peer Review](#)[Quick Links](#)» [Se](#)**Welcome to IEEE Xplore®**

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

Member Services

- ☐ Join IEEE
 - ☐ Establish IEEE Web Account
 - ☐ Access the IEEE Member Digital Library
- [Print Format](#)

Your search matched **2** of **989552** documents.A maximum of **2** results are displayed, **15** to a page, sorted by **Relevance** in **descending** order.

You may refine your search by editing the current search expression or entering a new one the text box.

Then click **Search Again**.

(parallel) and heap and (proces
multiprocessor or "multiple processor")">

Results:Journal or Magazine = **JNL** Conference = **CNF** Standard = **STD****1 A shared-memory multiprocessor garbage collector and its evaluation committed-choice logic programs***Imai, A.; Tick, E.;*Parallel and Distributed Processing, 1991. Proceedings of the Third IEEE Sympo
on , 2-5 Dec. 1991

Page(s): 870 -877

[\[Abstract\]](#) [\[PDF Full-Text \(716 KB\)\]](#) **IEEE CNF****2 Evaluation of parallel copying garbage collection on a shared-memory multiprocessor***Imai, A.; Tick, E.;*

Parallel and Distributed Systems, IEEE Transactions on , Volume: 4 Issue: 9 , 1993

Page(s): 1030 -1040

[\[Abstract\]](#) [\[PDF Full-Text \(960 KB\)\]](#) **IEEE JNL**

[Home](#) | [Log-out](#) | [Journals](#) | [Conference Proceedings](#) | [Standards](#) | [Search by Author](#) | [Basic Search](#) | [Advanced Search](#)
[Join IEEE](#) | [Web Account](#) | [New this week](#) | [OPAC Linking Information](#) | [Your Feedback](#) | [Technical Support](#) | [Email Alerting](#)
[No Robots Please](#) | [Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)

Copyright © 2003 IEEE — All rights reserved

IEEE HOME | SEARCH IEEE | SHOP | WEB ACCOUNT | CONTACT IEEE


[Membership](#) | [Publications/Services](#) | [Standards](#) | [Conferences](#) | [Careers/Jobs](#)
IEEE Xplore®
 RELEASE 1.5

 Welcome
 United States Patent and Trademark Office

[Help](#) | [FAQ](#) | [Terms](#) | [IEEE Peer Review](#)
[Quick Links](#)
[Welcome to IEEE Xplore®](#)
[SEARCH RESULTS](#)
[\[PDF Full-Text \(716 KB\)\]](#)
[NEXT](#)
[DOWNLOAD CITATION](#)

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account
- ☐ Access the IEEE Member Digital Library

[Print Format](#)

A shared-memory multiprocessor garbage collector and its evaluation for committed-choice logic progr

Imai, A. Tick, E.

Icot, Tokyo;

This paper appears in: Parallel and Distributed Processing, 1991.
Proceedings of the Third IEEE Symposium on

Meeting Date: 12/02/1991 -12/05/1991

Publication Date: 2-5 Dec 1991

Location: Dallas, TX , USA

On page(s): 870-877

References Cited: 14

IEEE Catalog Number: 91TH0396-2

INSPEC Accession Number: 4368135

Abstract:

A parallel copying garbage collection algorithm for symbolic languages executed on shared-memory multiprocessors is proposed. The algorithm is an extension of Baker's sequential algorithm with a novel method of heap allocation to prevent fragmentation and facilitate load distribution during garbage collection. An implementation of the algorithm within a concurrent logic programming system VPIM, has been evaluated and the results, for a wide selection of benchmarks analyzed. The authors show (1) how much the algorithm reduces the content of critical sections during garbage collection, (2) how well the load-balancing strategy works and its expected overheads, and (3) the expected speedup achieved by the algorithm.

Index Terms:

[logic programming](#) [parallel algorithms](#) [shared memory systems](#) [storage management](#) [VPIM](#) [benchmarks](#) [committed-choice logic programs](#) [heap allocation](#) [load-balancing](#) [parallel copying garbage collection algorithm](#) [sequential algorithm](#) [shared-memory multiprocessor garbage collector](#) [symbolic languages](#)

Documents that cite this document

Select link to view other documents in the database that cite this one.

[SEARCH RESULTS](#)
[\[PDF Full-Text \(716 KB\)\]](#)
[NEXT](#)
[DOWNLOAD CITATION](#)

[Home](#) | [Log-out](#) | [Journals](#) | [Conference Proceedings](#) | [Standards](#) | [Search by Author](#) | [Basic Search](#) | [Advanced Search](#) | [Join IEEE](#) | [Web Account](#) | [New this week](#) | [OPAC Linking Information](#) | [Your Feedback](#) | [Technical Support](#) | [Email](#)
[No Robots Please](#) | [Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)